A Disciplined Approach to Modernizing Advanced Software Systems

Dr. Charles D. Norton
NASA Jet Propulsion Laboratory MS 168-522
Imaging and Spectrometry Systems Technology Section (385)
High Performance Computing Systems and Applications Group
(818) 393-3920
nortonc@bryce.jpl.nasa.gov

Submission Approved by Dr. Thomas A. Cwik, Group Supervisor, High Performance Computing (385)

1. Problem Statement

JPL missions are constantly evolving, increasing the demands imposed on critical software systems and applications. This ambition requires more modern, flexible, reusable, and extensible component-based software that does not abandon the production applications required for success. Resolving this problem is vital to the future of software integration with mission planning at JPL.

In this proposal a general software design methodology is presented for modernization of legacy scientific and engineering applications. This technique leverages the existing investment in production codes while infusing modern software engineering principles. Our goal is to examine and evaluate various classes of mission flight software throughout JPL, such as flight, data, and analysis systems software on existing and new platforms, to determine how our technology can benefit such systems.

As an example, we propose to apply these ideas to the Modeling and Analysis for Controlled Optical Systems (MACOS) software important to the Next Generation Space Telescope (NGST) and Space Interferometry Mission (SIM) projects. MACOS, developed at JPL, provides tools for analysis of optical systems, model generation, and modeling for system-level design and analysis tasks.

2. Justification and Benefits with Estimated Return on Investment

Software engineering advances, such as component-based design and object-oriented programming, can improve the usability and management of sophisticated applications. These approaches can contain costs, improve quality, and support abstraction-based programming [1]. While object and component technology lack precise definition [1], most agree they fulfill a requirement while facilitating composition of software via specific interfaces. The technology, applied to static, dynamic, distributed, and/or non-distributed applications, encourages replaceable abstractions useful for complex, evolving systems [2-3]. Languages such as C++ and JAVA have some support for these ideas causing many to believe that existing Fortran-based scientific codes must be rewritten to benefit from these techniques. Rewriting mission critical applications, however, introduces extreme risk since it is time consuming, costly, the outcome is uncertain, and the benefits of experienced designers is potentially lost.

We have found that Fortran 90/95 has new features to support object-oriented principles beneficial for scientific programming [4-6] and, in this proposal, suggest a design methodology that defines a step-by-step process to modernize legacy application codes using state-of-the art software practices. While we emphasize Fortran applications, due to the abundance of legacy codes, similar techniques could be applied to software written in other domain-specific or general-purpose languages, including C or C++. As such, existing flight-related software is not excluded from modernization.

The benefits of our approach for improving the quality of mission software are numerous:

- Applications can be modernized while the software remains in productive use.
- Existing software remains unchanged, which is cost effective, as safety and extensibility are increased.
- Development in a new language is not needed preserving the designer's experience.
- Component-based design is applied promoting collaborative development.
- The methodology is based on well-defined standards that can be universally applied.
- Resolves the major problems exhibited in older codes such as lack of dynamic memory, abstraction, extensibility, collaborative development, inconsistencies, and safety.

Our process first upgrades the existing Fortran application to standard conforming Fortran 90/95. Next, interfaces to the original application routines are introduced to add safety features by detecting common programming errors. These interfaces also act as a boundary to routines used in creating an abstraction layer to the original, unmodified, code. This layer allows components to be introduced that interact cleanly with the original code, and that support new code enhancements.

Some companies claim to have tools that perform modernization from Fortran 77 to Fortran 90 automatically. An example is Simulog's Foresys tool (http://www.simulog.fr/iforef.htm). Unfortunately, such tools restructure in a non-human-readable manner (variable names are changed), do not add abstraction, and often leave software in a form nearly impossible to modify or extend.

In the MACOS project, the designers developed an effective code, but they have also realized that new mission requirements require a more flexible and abstraction-oriented code. Efforts were introduced to rewrite the code completely in C++, but the outcome was abandoned. Primarily, the new software did not perform as desired and the original designers are more fluent in Fortran.

The cost effectiveness can be quantified by multiple measures. Simultaneous retraining in a new language and programming style is eliminated. (Our experience is that software developers are reluctant to abandon experience gained over many years.) New development continues with modernization, eliminating delays, rather than consuming time in rewriting software from scratch. Errors in translation to a new language are eliminated. Also, the original investment in existing software is not lost. There are new concepts to learn, but they are in the context of a language one already knows, and these

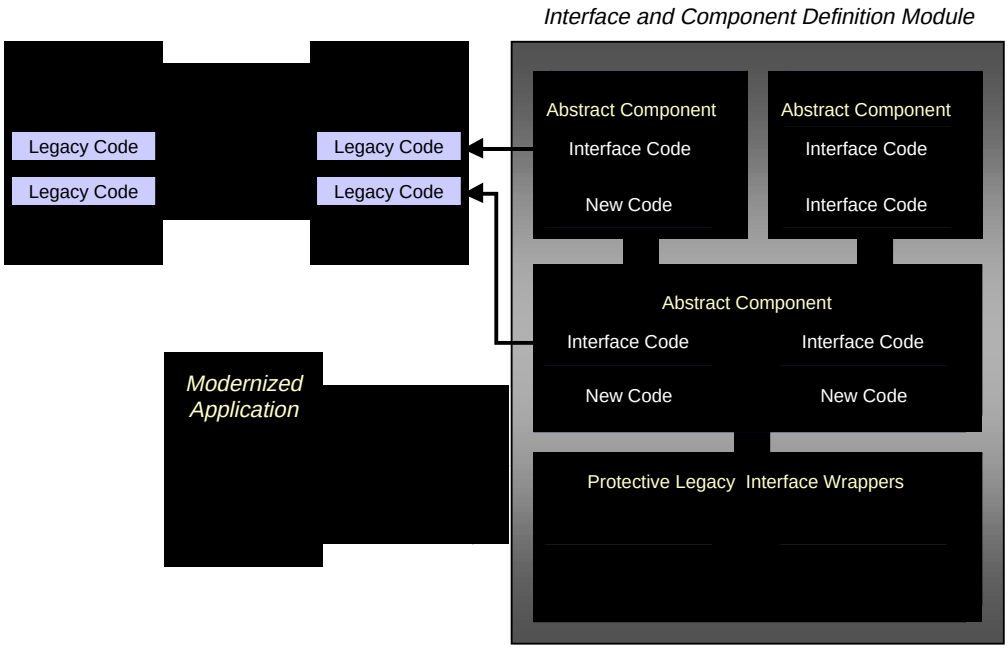
features are standardized. Using MACOS as an example the potential return on investment is large, especially when compared rewriting the software.

Projected Expenses for Rewriting Software (MACOS Example)			
Re-education	Software Rewrite	Original Code Cost	Total Expense
~$15,000.00	~$400,000.00	~$800,000.00	~$1,215,000.00

Estimating costs is complex as MACOS was created over an 8-9 year period where development and usage funding occurred hand-in-hand. Nevertheless, the potential return on investment (6 work months for this proposal) is high compared to the total expense of a software rewrite, using MACOS as an example. Any project that seeks to benefit from software modernization for scientific programming can calculate similar cost savings. Furthermore, the techniques we propose add to knowledge management of software systems positively impacting an entire project.

3. Workplan, Technical Approach, and Participation

Our approach promotes the principles of object-oriented design and component technology to modernize Fortran-based applications. When we speak of these concepts we mean abstraction-oriented software design for non-distributed systems [1].



Given the original procedure-oriented legacy software, it is reworked to use standardized, portable, Fortran 90/95 constructs. This includes replacement of common blocks with Fortran 90/95 modules, removal of implicit variable declarations, replacement of the non-portable `include` facility, and so on. This creates a standard compliant code that eliminates common errors and inconsistencies.

Next, Fortran 90/95 interfaces are written for the existing software, providing type checking and argument matching between the legacy code and the new abstract components that will subsequently be introduced. This stage often finds additional errors that can be corrected. This also forms a protective barrier to preserve the legacy code, although sometimes minor changes are needed (replacing `include` statements with Fortran 90/95 module `use` statements, for example). The interface code includes a wrapper routine (not shown) that allows existing code to be preserved, but a new context is formed for development and extension based on components. The wrapper is now the only way to communicate with the legacy code that has been standardized.

Abstract components can now be created by encapsulation (grouping) of related routines and associated interfaces. New code can also be added to the component framework for extensibility where supporting routines already exist, and they may interact. In MACOS, for example, optical elements with related properties can form one kind of reusable component that interacts with ray tracing routines defined in a separate component.

The modernized main program is reorganized to call routines defined by the new components. Depending on the amount of abstraction introduced, the main program may have significant changes. In the end, however, the new software remains familiar to the developers, but supports modern software practices for current and future development.

We have applied modernization techniques using object-oriented programming for existing work, such as the DOE sponsored Numerical Tokamak Project [7], and for new projects including Parallel Adaptive Methods for the NASA Earth and Space Sciences (ESS) project [8]. This proposal focuses on extending these ideas for legacy codes that must be modernized to achieve new JPL mission goals. Without this technology advances are hindered since the technical skill required might not be available under a new software plan and because there is uncertainty in changing one's experience.

Dr. Viktor K. Decyk (UCLA Department of Physics and Astronomy) has collaborated with the author on object-oriented scientific programming in Fortran 90/95 for years. As a faculty part time member of the High Performance Computing group at JPL he would participate in this work as a no-cost consultant.

4. Endorsements

Our work in applying object-oriented techniques using Fortran 90 has been recognized internationally and has been applied to projects at JPL including NASA's ESS Project and the Microwave Limb Sounder (MLS) flight project. This proposal to extend the work toward a formal process for modernization of legacy scientific applications has been endorsed by the following.

"NGST, SIM and many other projects use MACOS for critical design and engineering computations. It has provided JPL with a unique capability that has paid off in our efforts to win new flight projects. The continued growth and improvement of MACOS will result in improved efficiency and effectiveness for our technology tasks and help continue our leadership in the controlled optics and integrated modeling fields."

Dr. David R. Redding
NASA Jet Propulsion Laboratory, MS 306-438
Optical Systems Modeling (385)
(818) 354-3696
dcr@huey.jpl.nasa.gov

"I fully support this proposal. My experience with using your F90/F95 object-oriented methodology indicates that developing and maintaining scientific applications in F90/F95 is highly efficient and reliable. This methodology has definitely contributed to our success in the development of CATO (Computer Algorithm for Trajectory Optimization) in Section 312. It has been my experience that choosing the programming language that best fits the application is the key to successful software engineering applications. Legacy software, particularly scientific applications, has built in expertise and hidden knowledge that took years to accomplish. Therefore, writing C++ or Java "wrappers" around legacy software is a safe and cost effective means of preserving this expertise and hidden knowledge."

Mr. Jack N. Hatfield
NASA Jet Propulsion Laboratory / ACRO Service Corporation, MS 301-140R
Mission Execution and Automation (368)
Senior Software Engineer, Mission Analysis Software Team
(818) 354-2198
Jack.Hatfield@jpl.nasa.gov

"The Interferometry Center of Excellence fully supports this proposal. We are currently providing financial support for the development of both the IMOS and MACOS analysis tools, as we see their availability as essential to the analysis and development of current and future expected interferometry missions. MACOS is the optical modeling module which is utilized by the IMOS (Integrated Modeling of Optical Systems) tool. The proposed modernization and wrapping of the MACOS code will greatly improve its usability, safety and utility."

Mr. Willis C. Goss
Deputy Leader, Interferometry Center of Excellence (Organization 302), MS 301-486
(818) 354-4540
Willis.Goss@jpl.nasa.gov

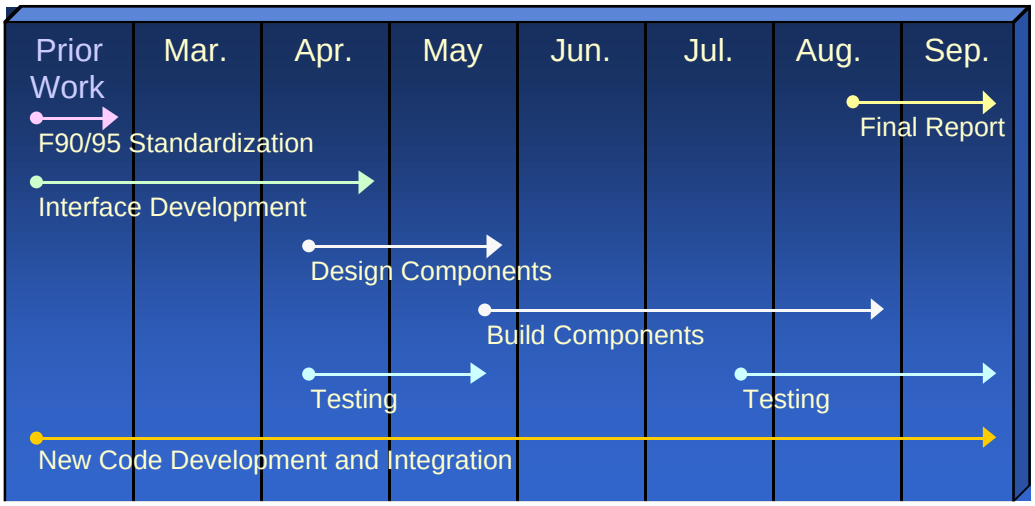
"It is clear to me that substantial legacy FORTRAN code still exists in the science modeling, simulation, and engineering activities at JPL. This proposal addresses essential issues regarding the continuing efficient use of this legacy software."

Dr. Robert D. Ferraro
NASA Jet Propulsion Laboratory, MS 180-604
Manager, Remote Exploration and Experimentation Project
Associate Project Manager, NASA Earth and Space Sciences Project
Technology and Applications Program Directorate
(818) 393-5269

ferraro@zion.jpl.nasa.gov

5. Schedule and Deliverables

- MACOS contains approximately 60,000 lines of Fortran 77/66 and will be used as the main testbed for our work. The main deliverable will be a final report describing the modernization methodology and an overview of how it can potentially be applied to other flight system software at JPL.



The schedule outlines the development plan for MACOS, where work prior to the proposal start date is shown. Although actual software will be written for MACOS, the report will also focus on our experiences in examining other prototypical software at JPL.

6. Cost

The costs, broken down by JPL labor, procurements, travel, and services are listed in the proposal budget table below.

Personnel	Funded Work Months
Charles D. Norton	6
Proposal Cost Items	**Proposal Cost Plan**
Salary	Requested
Procurements	Not Requested
Travel	Not Requested
Services	Not Requested

7. Sources of Co-Funding and/or Follow-on Funding

Sources of funding and co-funding are:

- Interferometry Center of Excellence (ICE) funded standardization to Fortran 90/95, resulting in MACOS version 2.9α, released 12/99.
- SIM Interferometry Technology Project is funding additional, optical modeling improvements for incorporation into MACOS v2.9 (to be completed 9/00).

8. Conclusion

We presented a design method that modernizes critical production software. Reliable applications, that are beginning to limit new mission objectives, can be refurbished to achieve these goals at tremendous cost savings. The functionality is preserved and collaborative development becomes possible. Indeed, the modern design should simplify plugging applications into new environments and interaction with object-oriented component-based codes in other languages.

Incidentally, the need to resolve these issues are critical to other NASA programs as well. Round III of the NASA Earth and Space Sciences (ESS) Project (http://esdcd.gsfc.nasa.gov/ESS/CAN2000/CAN.html) is focused on improving the software engineering aspects of scientific applications. This is a multi-year multi-million dollar program. Our goal is to demonstrate, and later infuse, this technology into the JPL software development process as a common practice.

9. References

1. Brown, A. and Wallnau, K. C. The Current State of CBSE. *IEEE Software*, 15(5):37-46, September/October 1998.

2. Kozaczynski, W. and Booch, G. Component-Based Software Engineering. *IEEE Software*, 15(5):34-36, September/October 1998.

3. Meyer, B., and Mingins, C. Component-Based Development: From Buzz to Spark. *IEEE Computer*, 32(7):35-37, July 1999.

4. Decyk, V. K., Norton, C. D., and Szymanski, B. K. How to support inheritance and run-time polymorphism in Fortran 90. *Computer Physics Communications*, 115:9-17, December 1998.

5. Decyk, V. K., Norton, C. D., and Szymanski, B. K. How to Express C++ Concepts in Fortran 90. *Scientific Programming*, 6(4):363-390, Winter 1997. IOS Press.

6. Norton, C. D., Decyk, V. K., and Slottow, J. Applying Fortran 90 and Object-Oriented Techniques to Scientific Applications. In S. Deymer and J. Bosch, editors, *Object-Oriented Technology*, pages 462-463, Brussels, Belgium, July 1998. ECOOP Conference Workshop Reader, Springer LNCS 1543.

7. Norton, C. D., Szymanski, B. K., and Decyk, V. K. Object-Oriented Parallel Computation for Plasma Simulation. *Communications of the ACM*, 38(10):88-100, October 1995.

8. Lou, J. Z., Norton, C. D., and Cwik, T. A Robust and Scalable Library for Parallel Adaptive Refinement on Unstructured Meshes. In C. Schulbach and E. Mata, editors, *NASA ARC Comp. Aerosciences Workshop 98*, pages 241-246, January 1999.